Optimization Principles for Codesign applied to Molecular Dynamics: Design Space Exploration, Performance Prediction, and Optimization Strategies

Stephan Eidenbenz (PI, POC, eidenben@lanl.gov), Kei Davis (Co-PI), Art Voter (Co-PI), Hristo Djidjev, Leonid Gurvits, Christoph Junghans, Susan Mniszewski, Danny Perez, Nandakishore Santhi, Sunil Thulasidasan

Los Alamos National Laboratory
PO Box 1663
Los Alamos, NM 87544

Keywords: *Molecular Dynamics, Optimization, Computational Codesign, Hardware/Software Design Space Exploration*

Executive Summary

The objective of the research project *Optimization Principles for Codesign applied to Molecular Dynamics* at Los Alamos National Laboratory is to develop methodological and software frameworks for *hardware-software codesign as a formally posed optimization problem*.[1] While the optimization framework will be applicable to multiple problem domains, for the target application we use *molecular dynamics (MD),* an exemplar for the need for computational scaling (weak, strong, and combinations of both), and archetypical of the obstacles thereof. We view codesign as search and selection from a vast space of hardware and software designs that map to performance metrics. The objective functions that we optimize for have components such as *run time (or computational rate), problem size, simulated time duration, energy use,* and *hardware cost*.

Our semi-formal codesign optimization framework relies on

1. **Efficient enumeration methods** for finding feasible hardware architectures and software designs;
2. **A multi-scale approach to performance prediction modeling**, where we use cycle-accurate virtual machine emulation, discrete event simulation, graph mapping, and constraint programming as different prediction methodologies; and
3. **Optimization methods with fast identification of new hardware-software pairs** to be tested with the more detailed performance prediction methods.

We define software designs in a hierarchical fashion with the main hierarchy levels of Application/Problem, Model/Equations, Algorithms, and Software Implementation, where changes to higher levels are more fundamental than lower level changes of, say, a specific data structure. Enumeration of software designs is performed by efficient search methods with the potential help of a human for feasibility verification. The hardware architecture is defined as a hierarchical set of components that interact with each other and offer services to an application through operating system and system software; enumeration of architectures is done implicitly through parameter spaces and optimization search methods as well as (human, possibly machine-aided) design feasibility checking. The performance prediction method will choose its software design and a hardware description level that are of appropriate fidelities.

Our performance prediction methods model at different levels of detail, thus covering the space of trade-offs between accuracy and scalability in both time and size. Optimization heuristics are necessary for the more detailed prediction methods of virtual machine emulation and discrete event simulation. Our framework allows for the adaptation of standard meta-heuristics such as simulated annealing, tabu search, gradient search, and genetic algorithms. The more coarse-grained prediction methods of graph mapping and constraint programming will optimize through graph algorithms and mathematical programming techniques.

[1] This project is funded by a competed LANL Laboratory Directed Research and Development (LDRD) Directed Research (DR) award number 20120038DR.

From our chosen applications domain perspective, we will develop atomistic simulation tools that will enable the study of processes such as *ductile spall failure* under shock conditions and the *evolution of radiation damage*. Achieving this requires a two order of magnitude increase in simulated time over current state-of-the art petascale computing, and more importantly cannot be realized without this codesign approach, as direct implementation on an exascale platform cannot achieve this. These two applications address issues of the response of materials in extreme conditions and enabling the design of more effective and safe fission power plants, respectively.

Hardware-software codesign, perceived as a prerequisite to achieving exascale computing, needs to be put on a sound scientific basis such that design decisions for both hardware and software do not need to be made based on colloquial heuristic insights, but rather follow an established scientific procedure by a sufficiently thorough search of realizable hardware and software options.

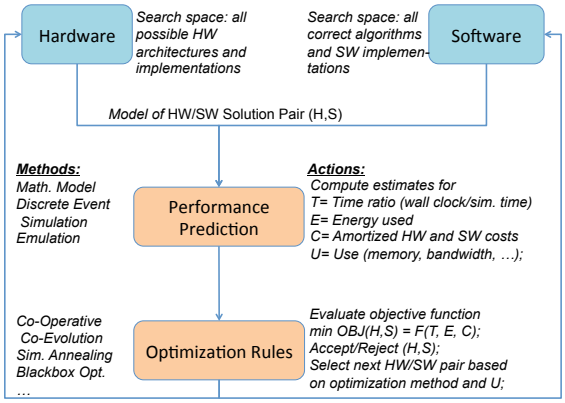

Fig. 1: Iterative CoDesign Optimization Loop

Fig. 1 illustrates our approach, where the left hardware design and the right software design boxes define a vast space of hardware and software solutions, whose combinations lead to performance prediction, the results of which in turn guide an optimization method towards new hw/sw solutions to be tested. The resulting iterative process can be analyzed in formal and informal settings, thus opening doors to established optimization and analysis techniques, while at the same time incorporating sometimes superior but informal human ingenuity. The key research efforts in this approach are

1. Efficient enumeration methods for the both the hardware and software design spaces ("What are our options?") that accurately reflect physical constraints and trade-offs in realizable hardware and software designs;
2. Performance prediction methods ("How fast will it run?"); and,
3. Optimization methods to search the design spaces ("How do we select solutions?").

Background: The Scaling Challenge

After many years of Moore's Law processor speeds have leveled out and advances in computational power are now coming from massive parallelism and architectural heterogeneity. In almost every scientific field, without radical modification of algorithms to adapt them to the hardware, harnessing the full theoretical speed of next-generation computers will be impossible. As a specific example, consider the application focus of this project, molecular dynamics (MD), a standard workhorse in materials science and many other fields. Given the linear scaling of MD with respect to the number of atoms, system size and total simulation time can in principle be traded for one another given a perfectly scaling MD code. Using a traditional parallel MD code based on spatial decomposition, i.e., where contiguous subsets of atoms are assigned to each processor, this ideal scaling is compromised by two factors: first, for very large numbers of atoms per node and hence short simulation times, the total system size is limited by the available memory; and second, for small numbers of atoms per node, the communication overhead required to pass the positions of the atoms across the interfaces between domains severely limits the achievable parallel efficiency. These two effects can be seen in the shape of the dark blue region in Fig. 2, which indicates simulations possible in 24 hours on the Petaflop Roadrunner machine [10] for a 1000-atom silver nanowire system. If the full capability of Roadrunner could be efficiently used for any size-time combination, the complete area under the petascale dashed line would be accessible.

For infrequent-event systems, which make only occasional transitions to a new state (e.g. ,the diffusive jump of a vacancy in a solid), accelerated molecular dynamics (AMD) methods developed at Los Alamos [4-7] can be employed to reach longer time scales. The AMD methods, which are described later, are typically most

effective for small systems of up to ~10^4 atoms. In the case of the parallel-replica dynamics (ParRep) method, many replicas of the entire system are evolved simultaneously and independently, eliminating the need for inter-domain communication except when a transition occurs. ParRep gives exact state-to-state dynamics if it is implemented carefully, and this re-casting of the work amounts to parallelization of time. The chief overhead in ParRep is associated with the preparation of the new replicas after each transition, so for a system with very infrequent events, long-time simulations that come close to the right end of the theoretical simulation regime are possible, as shown by the line bounding the green region in Fig. 2. This curve assumes a transition rate proportional to system size, a typical behavior. Note that in spite of the impressive time scale that can be reached with ParRep, there remains a substantial mid-range region under the dashed curve that cannot be reached with any current MD simulation method, on current hardware, for this type of problem.

The severity of the problem facing us becomes clearer when we estimate the behavior of MD or ParRep on an anticipated realization (in say 2020) of an exascale machine with 10^8 processing cores (Fig. 2). The increased memory will allow simulations up to ~10^{14} atoms for ps time scales, and ParRep for very small systems might reach second time scales if the events were very infrequent (once per 10 ns for 1000 atoms), but essentially no new territory is covered in the middle range. The important regime of 10^6 to 10^9 atoms for microseconds to milliseconds is still entirely inaccessible, meaning we will have no direct way to simulate processes such as ductile spall failure, dislocation pileup and release at grain boundaries, or radiation damage annealing involving multiple cascades, all of which are important materials science problems and critical to the DOE mission. *In short, if we do not find a new paradigm for implementing MD methods on advanced computer architectures, critical problems in materials science that could in principle be addressed in a powerful way with MD will remain utterly out of reach for the indefinite future.* We believe the answer to this problem lies in the development of a formal codesign capability that optimizes hardware and software simultaneously.

Ideally a codesign effort would enjoy unrestricted latitude in the entire spectrum of the design space, from methods, algorithms, to hardware, and in fact domain-specific, hardware-level-up codesigns have been realized, notably MDGRAPE-3 [3] and D.E. Shaw Research's Anton [2], both for molecular dynamics calculations on biological systems.

In general, chip-level design will be market driven. We posit that even absent the ability to create custom hardware, predictive capability across the entire codesign spectrum is still important. First, it makes it possible answer the questions "What could be done in principle?" as well as "What can be realized in practice?" Second, it can stimulate fundamental algorithmic reformulation for existing and anticipated hardware architectures, as demonstrated for example by early work at D.E. Shaw Research [1]. We note that some influence by the HPC community in processor design is possible, e.g. the eDP variant of the IBM Cell BE for LANL's Roadrunner. Given a sound predictive capability, arguments to hardware designers for architectural changes become much more compelling.

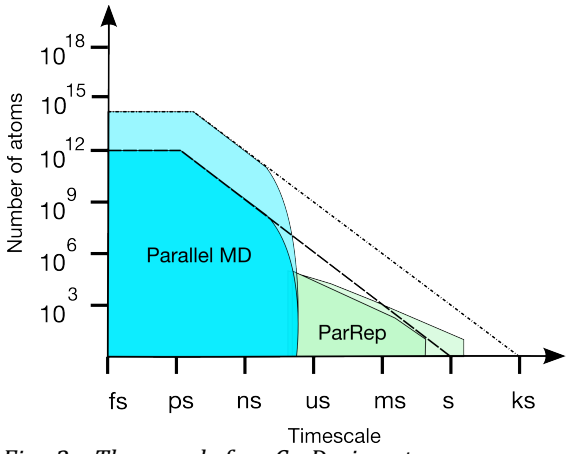

Fig 2: The need for Co-Design to overcome current performance boundaries of MD/AMD. Strongly colored regions correspond to the accessible problem space on a current petascale computer (24 hours on Roadrunner) while paler regions are for an exascale machine. The dashed and dash-dotted lines correspond to the theoretical limit of a perfectly-scaling MD code, at the peta- and exascale, respectively. Note the capability hole at 10^6-10^9 atoms for μs to ms.

A successful codesign optimization framework will fundamentally change the way scientific applications are constructed: HW/SW codesign fundamentals will become an integral part of software engineering. The optimization framework will have manual and automated aspects, similar to compiler optimization techniques

used in classical compilers for the automated aspects; it will achieve peak performance when the human designer allows the optimizer to test a large set of potential solutions in a high-throughput optimization loop.

Technical Approach: Optimization Principles in Codesign

Our semi-formal codesign optimization framework optimizes relies on

1. **Efficient enumeration methods** for finding feasible hardware architectures and software designs;
2. **A multi-scale approach to performance prediction**, where we use cycle-accurate emulation, discrete event simulation, constraint programming, and closed-form mathematical modeling as different prediction methodologies; and
3. **Optimization methods with fast identification of new hardware-software pairs** to be tested with the more detailed performance prediction methods.

We define hardware and software designs in a hierarchical fashion. Enumeration of software designs is initially by parameterization of known algorithms, subsequently we will explore correctness-preserving transformations and compositional methods. The enumeration of hardware architectures is done implicitly through parameter spaces and optimization search methods as well as (human, possibly machine-aided) design feasibility checking.

Our performance prediction methods model at different levels of detail, thus covering the trade-off space of accuracy vs. scalability in both time and size. Optimization heuristics are necessary for the more detailed prediction methods of virtual machine emulation and discrete event simulation. Our framework allows for the adaptation of standard meta-heuristics such as Metropolis walk, simplex, simulated annealing, tabu search, gradient search, and genetic algorithms.

Guided by our application of choice, molecular dynamics (MD) and in particular Accelerated Molecular Dynamics (AMD) methods, we have divided our concrete efforts in building an optimization loop in an hour-glass fashion at the computational core of any MD method, namely the force call. Recall that force call computations compute the force acting on each atom in the MD simulation as given by a potential function that needs to be evaluated. The logic of an AMD method can be viewed as minimizing the number of force calls that are to be made while exploiting available computational resources in terms of computational cores as best as possible. We thus aim to define optimization loops that work best at either optimizing the AMD method or optimizing the force call. It is a fortunate and perhaps philosophical observation that the level of detail required to model the AMD logic is that of a computational unit (e.g., single core, multiple cores) and communications media (e.g., on-chip or interconnect network) connecting those units, while a model of the force call code requires details of cache hierarchies, pipelining, data motion, and other board-level details. Our main task of enumerating and pruning the hardware and software search spaces to a reasonable complexity thus naturally comes in two incarnations that still can borrow some insights from each other.

The performance prediction methods for the AMD logic are currently mathematical models and discrete event simulation on an AMD state level, whereas the force call details are better modeled with near-cycle-accurate emulation. Since both ends of the hourglass result in a number of parameters to be optimized, we can build optimization strategies that work for both sides. Along these lines we distinguish two approaches on the AMD logic level:

1. Performance models in closed mathematical form using Markov chain theory. While the main goal has been to analyze performance, we have serendipitously found a promising new AMD method that has the potential to be very efficiently parallelized.
2. AMD logic state simulation relies on a fast-running skeleton mini-app implemented as a scalable discrete event simulation model. This application, AMDSim, operates at the algorithmic level and is described below. In the force call analysis method, we have designed and partially implemented an approach we call Profile, Parallelize, and Predict that we describe below. It prunes the search spaces by building a partial-order execution graph of a force call code, identifies parallelism opportunities that it then uses to build an optimum hardware solution with as much parallelism as possible. Finally it predicts performance by simulating the code on the custom-designed hardware using our Architecture Simulation and Exploration Tool (ASET). We also use a provably optimum low-level

circuit design method based on solving satisfiability problems that we use as building blocks of the architectures fed to ASET.

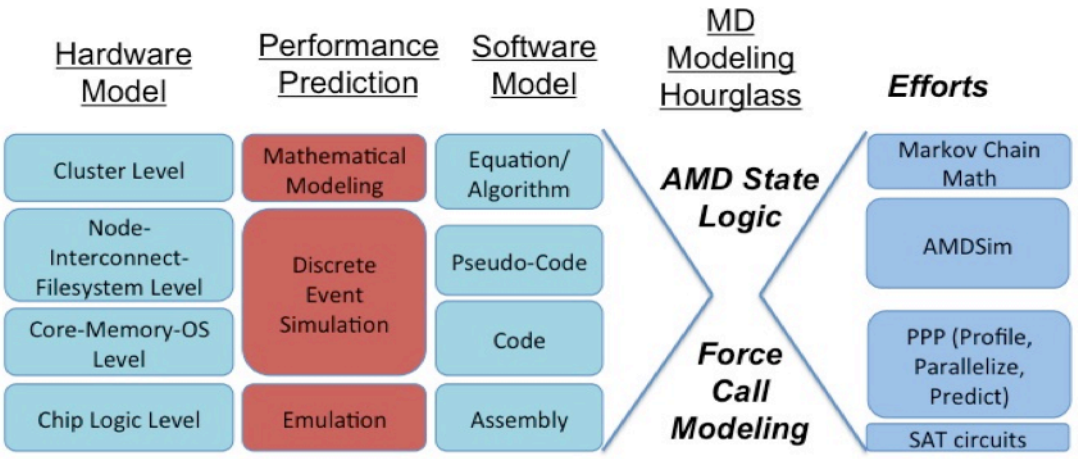

Fig. 3: Different levels of detail for modeling the hardware side and the software side usually have a natural choice for the performance prediction method. Our concrete modeling efforts shown on the right fall onto either side of the MD modeling hourglass that distinguishes between AMD State Logic and the details of the force call.

Upper Hourglass Half: AMD State Logic Modeling

Our initial plan included a mathematical performance prediction component that would provide analytical predictions of the performance of AMD methods from first principles. The rationale was to allow for rapid exploration of parameter spaces that would assist in the optimization. This approach can indeed provide useful guidelines but closed form expressions can only be derived under rather strict assumptions. These limitations highlighted the need for a flexible performance prediction methodology that is now a cornerstone of our approach. The deep mathematical analysis of the algorithms did however yield novel insights into the nature of the methods and allowed for the derivation of variants with different algorithmic properties. For example, we showed how TAD could be reformulated as a Metropolis Monte Carlo-based method that can be more easily parallelized than the original version. Further investigation of the performance of this new algorithm is now underway with initial documentation available [8].

This lack of amenability to analytic performance modeling, even at a fairly high level of abstraction, somewhat serendipitously provided an opportunity to quickly demonstrate the feasibility and practical utility of the more general optimization loop approach early in the project. We devised relatively simple `software spaces'—essentially parameterized models of the MD algorithms as state machines—in which we could cast meaningful optimization problems. These formulations mimic the AMD algorithms with the force calls replaced by (simulated) time delays. The optimizations they enable are of considerable practical interest to the molecular dynamicists and do not have solutions that can be intuitively guessed even by seasoned practitioners. As previously described, and depicted in Fig 2, the major challenges in MD are in achieving greater system-size scales and greater simulation-time scales. The former, the weak-scaling problem, can in principle be solved with more hardware capacity—more memory and processors. The latter, the strong-scaling problem, has been, and continues to be, the subject of considerable research. With current hardware technology, for a class of MD systems of interest, conventional MD can achieve about 1 microsecond simulation time for reasonable wall-clock application run time, but the relevant time scale for exploring meaningful system dynamics is often much longer—milliseconds, seconds, or longer. For such *infrequent event systems* LANL researchers have developed *accelerated molecular dynamics* (AMD) methods that can reach much longer time scales than conventional methods. Two such approaches are the *temperature accelerated dynamics* (TAD) methods and the *parallel replica* (ParRep) methods. Significantly, these approaches are orthogonal in the sense that they may be synergistically combined into hybrid methods.

The idea behind temperature-accelerated dynamics is to greatly accelerate the rate of significant events by simulating the system at high temperature. Every attempted event is intercepted, its time is extrapolated to low temperature, and accepted only if it would have occurred first at low temperature. TAD yields long time dynamics that is accurate at the level of harmonic transition state theory. The optimal high temperature for a system results in the shortest wall clock time for execution.

While it is empirically known that for a number of systems and ranges of temperatures of interest with increasing temperature the gain outweighs the overhead up to a point, beyond which the overhead dominates, the exact shape of this curve is not generally readily expressed analytically. However, for the purpose of finding the optimal temperature elevation, it is not necessary to perform the actual MD simulation; it is sufficient to simulate only the higher-level mechanics of an MD simulation while consuming a small fraction of the compute resources. In this scenario we are optimizing a single quantity—the *real boost*—with respect to approximately a dozen variables describing the system.

The ParRep method, in contrast to TAD, achieves acceleration by simulating multiple instances, or replicas, of the system in parallel. Given that true infrequent events have an exponential first-passage time distribution, we can exploit this to parallelize time, by having multiple replicas seek the first escape event. Statistical independence of the replicas is achieved by appropriately randomizing or *dephasing* their initial states. Arbitrarily accurate dynamics are possible when implemented carefully.

Because of the constant (per replica) overhead of *dephasing*, parallel efficiency decreases as the number of replicas increases for a given system, but the computational boost still increases. Algorithmic choices, physical system characteristics, and hardware characteristics can be varied. Multiple trials for a given system and set of parameter choices run in a small fraction of time and compute resources required for actual ParRep simulation. Parameters can be chosen to optimize the computational boost. Hybridizing ParRep with TAD will provide another dimension over which to optimize—a temperature/parallelism tradeoff.

These models have been implemented as parallel discrete event simulations of MD simulations. This suite, dubbed AMDSim, uses the SimCore parallel discrete event simulation framework [9]. To date the TAD simulator, TADSim, has been verified against experiment, and the ParRepSim simulator has been demonstrated with up to one million replicas. In due course these will be combined to provide performance prediction for the hybrid AMD methods.

Thus far the hardware characteristics are implicit, characterized by performance metrics (e.g. computation time for a single force call) provided as inputs to the simulations. This design does, however, lend itself to direct augmentation with increasingly detailed spaces of hardware models using the various techniques that span our intended spectrum of time vs. accuracy tradeoffs.

Lower Hourglass Half: Force Call Modeling

We make use of several new techniques in performance prediction for HW/SW co-design at the behavioral level, timed transaction level, down to cycle-accurate and gate levels. The primary objective for a virtual-ware based performance prediction is to obtain high fidelity, timed transaction level model (TLM) resource usage estimates. Timed TLM is preferred as it can capture adequate detail of the software and hardware components, while not being too detailed.

An iterative optimization based hardware/software co-design flow

Scientific applications such as physics codes are often highly repetitive and parallelizable, such that speedups of several orders of magnitude may be possible with optimized, custom hardware/software solutions. For arriving at an optimal hardware/software pair, we have devised a greedy *profile-parallelize-predict* (PPP) iterative hardware/software co-design and optimization flow, as follows.

1. The user application is first converted into a representative *MiniApp* in a high-level language such as C. The minimal app is profiled at run-time on typical data to identify hotspots for optimization.
2. Identify inherent parallelism, prioritizing hotspots.
    a. MiniApp based static analysis is used to obtain Program Dependence Graphs (PDG) of hotspots.

   b. Form a partial order directed acyclic graph from the PDG by condensing its strongly connected components, capturing both control flow and data dependency in a single graph. Optionally variable assignments on the PDG are represented in a *Single Static Assignment* (SSA) form, to simplify hardware memory block allocations by removing global data dependencies in the prototype hardware architecture.
   c. Construct a prototype architecture including memory and processing elements based on the partial order graph.
   d. Specify architecture in low-level assembly language directives that ASET understands.
3. Perform data motion analysis to identify bottlenecks in RAM/Cache/ROM access using.
   a. Compile the high-level C code to a simplified intermediate language and then to the prototype architecture directives.
   b. Simulate the prototype architecture assembly using the ASET prototype architecture emulator with real data, and realistic memory models incorporating ROM, RAM, cache-stages and registers.
   c. Identify typical data-motion costs and bottlenecks from prototype emulation.
4. Iteratively modify the algorithm and the prototype architecture based on partial-order graph parallelism and emulated data motion analysis. Using the ASET performance predictor, the hardware space can be heuristically searched using for instance a simulated annealing approach for a locally optimal configuration in terms of speed, number of gates, power-consumption, etc. The software space is also simultaneously searched using other application-specific techniques to identify optimal configurations of parameters.
5. Implement the identified optimal hardware design based on the prototype in a Hardware Description Language (HDL), and synthesis it into a gate level net-list.
   a. Optimize costly hardware modules using gate-level optimization through a SAT formulation.
6. Realize the finalized optimal architecture in FPGA or ASIC.

Tools for hardware performance prediction

ASET allows us to rapidly prototype hardware through cycle accurate emulation to predict run-time data-motion costs. ASET understands a set of standardized hardware modules. Other specialized modules can be implemented as needed. A prototype architecture will be composed of some or several of these standardized modules. An algorithm running on this prototype architecture is described as a sequence of low-level directives that are interpreted and emulated by the ASET simulation engine.

ASET lets us quickly assemble a prototype hardware architecture with custom timings for each processing directive, data-access directive, and communication directive. It can emulate several RISC processor nodes in a distributed memory architecture. Each processor has a generic, vectorized RISC assembly instruction set, and understands cache hierarchies, RAMs, stacks, and registers.

The ASET prototype architecture directives are similar to conventional assembly instructions, and are suitable for being targeted by a specialized LLVM back-end. ASET can emulate a networked topology, with message-passing communication among processor nodes.

Tools for gate-level optimization

Occasionally prototype hardware architectures include specialized modules that implement certain steps in an algorithm, for example a square-root calculation. Standard net-list synthesis often results in sub-optimal gate-level hardware circuits in such cases, whereupon it becomes desirable to synthesize optimal gate level hardware blocks subject to functionality, allowed gate types, delay, and parallelism constraints. We first identify algorithmic steps that are costly to implement in hardware. As an initial strategy, we try to optimize the algorithm and data-motion paths to avoid these costly steps. If a costly step turns out to be inevitable we synthesize an efficient hardware circuit at gate level to implement it.

We have developed a set of algorithms and tools in collaboration with a team at Claremont Graduate University (CGU) that implements a combinatorial gate-level optimization strategy. The hardware constraints such as latency, circuit-depth/width, allowed gate types, and module functionality truth table are all encoded into a single *Combinatorial Satisfiability Equation System* (SAT) model. This SAT system, which can be NP-hard to solve in general, is then heuristically solved using a standard SAT solver or a custom approach such as simulated annealing, a genetic algorithm, or linear-programming relaxation.

Conclusions

We have demonstrated the practical utility of using automatic optimization techniques as drivers for design in the algorithmic space, at least within the molecular dynamics application domain, though the techniques used are in no way specific to molecular dynamics. Demonstrations of hardware generation and optimization will follow very shortly. What remains, then, having surmounted this conceptual hurdle, is the bulk of the actual work, namely the extension and refinement of the software and hardware enumeration methods, the implementation of corresponding performance prediction methods, and the use of a broader set of optimization techniques over these spaces.

The hourglass model of distinguishing the computational kernel, such as the force call in the case of AMD, from the higher-level logic of a computational code is applicable to many application domains. Our current efforts of AMDSim and PPP aim to deliver insights in terms of codesign methods beyond AMD specifics. During the first year of this multi-year project, the focus has been on the first two tasks (sw/hw space enumeration and performance prediction) because these parts simply need to be in place before we can turn our attention to the third task of implementing more sophisticated optimization search strategies. We have begun implementing first optimization rules and plan to focus more intensely on this core part of the project in the remaining years.

References

[1] Kevin J. Bowers et al., Scalable algorithms for molecular dynamics simulations on commodity clusters. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing* (SC '06). ACM, New York, NY, USA, Article 84. DOI=10.1145/1188455.1188544 http://doi.acm.org/10.1145/1188455.1188544, 2006.

[2] David E. Shaw et al., Anton, A special-purpose machine for molecular dynamics simulation. In *Proceedings of the 34th annual international symposium on Computer architecture* (ISCA '07). ACM, New York, NY, USA, 1-12. DOI=10.1145/1250662.1250664 http://doi.acm.org/10.1145/1250662.1250664, 2007.

[3] Taiji M. MDGRAPE-3 chip: a 165 Gflops Application Specific LSI for Molecular Dynamics Simulations, *Proceedings of Hot Chips 16*, IEEE Computer Society

[4] A. F. Voter, "Hyperdynamics: Accelerated Molecular Dynamics of Infrequent Events", Phys. Rev. Lett. 78, 3908 (1997).

[5] A. F. Voter, "Parallel replica method for dynamics of infrequent events", Phys. Rev. B 57, R13985 (1998).

[6] M. R. Sorensen and A.F. Voter, "Temperature-accelerated dynamics for simulation of infrequent events, J. Chem. Phys. 112, 9599 (2000).

[7] D. Perez, B.P. Uberuaga, Y. Shim, J. Amar, and A.F. Voter, "Accelerated Molecular Dynamics Methods: Introduction and Recent Developments", Annual Reports in Computational Chemistry 5, 79 (2009).

[8] L. Gurvits et al., "Simulating one Markov Process based on a Black Box for another: Rigorous Proofs", manuscript, March 2012.

[9] S. Thulasidasan, S. Eidenbenz, S. Mniszewski, et al, "Designing systems for large-scale, discrete-event simulations: Experiences with the FastTrans parallel microsimulator", IEEE HiPC 2010.

[10] T. C. Germann, K. Kadau, S. Swaminarayan, "369 TFlop/s molecular dynamics simulations on the the petaflop hybrid supercomputer 'Roadrunner'," Concurrency Computat.: Pract. Exper. **21**, 2143-2159 (2009).